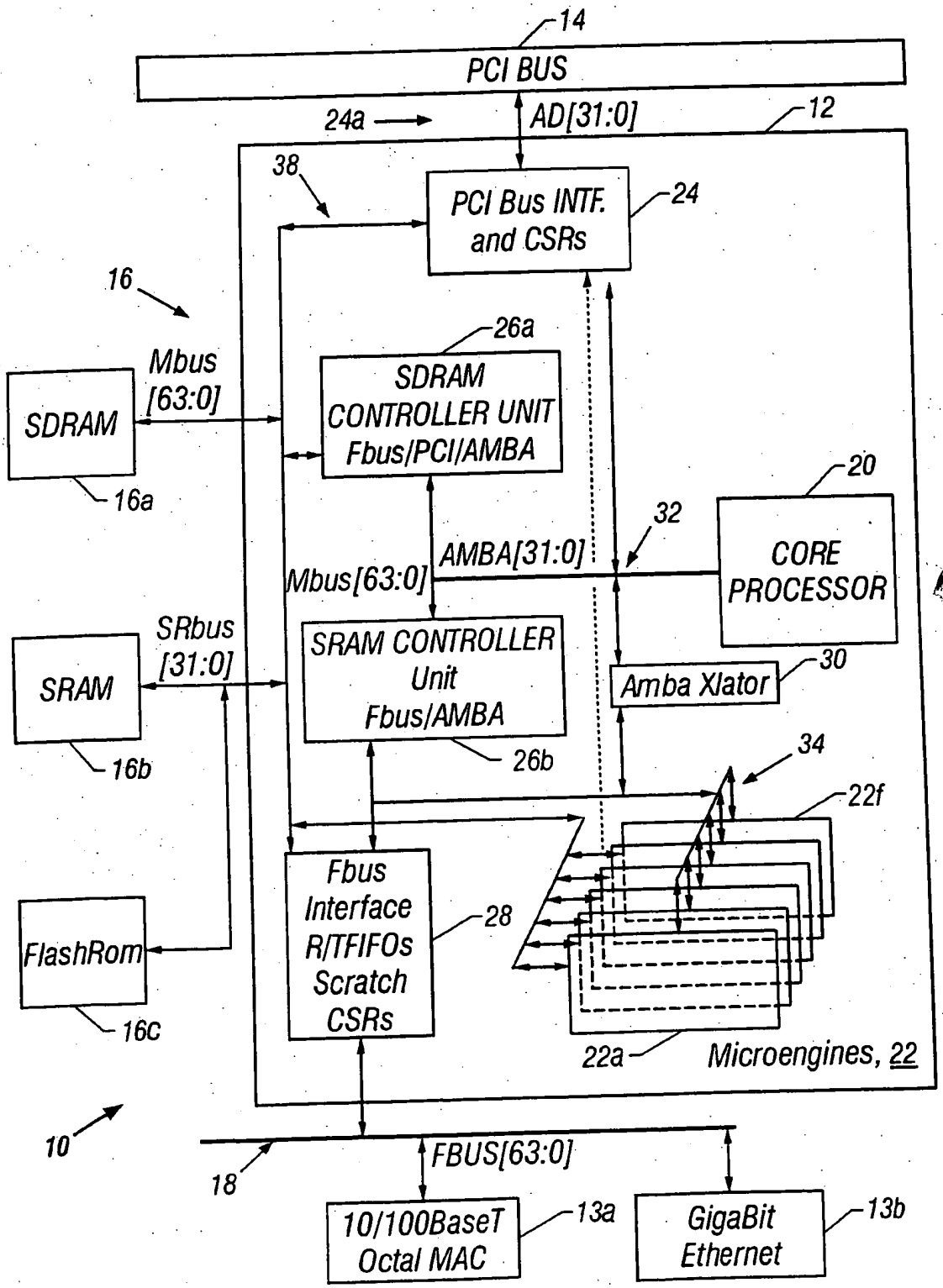

14
PCI BUS
24a
AD[31:0]
12
38
PCI Bus INTF. and CSRs
24
16
26a
Mbus [63:0]
SDRAM
SDRAM CONTROLLER UNIT Fbus/PCI/AMBA
16a
20
32
AMBA[31:0]
CORE PROCESSOR
Mbus[63:0]
SRbus [31:0]
SRAM
SRAM CONTROLLER Unit Fbus/AMBA
Amba Xlator
30
16b
26b
34
22f
Fbus Interface R/TFIFOs Scratch CSRs
28
FlashRom
16c
22a
Microengines, 22
10
18
FBUS[63:0]
10/100BaseT Octal MAC
13a
GigaBit Ethernet
13b

FIG. 1

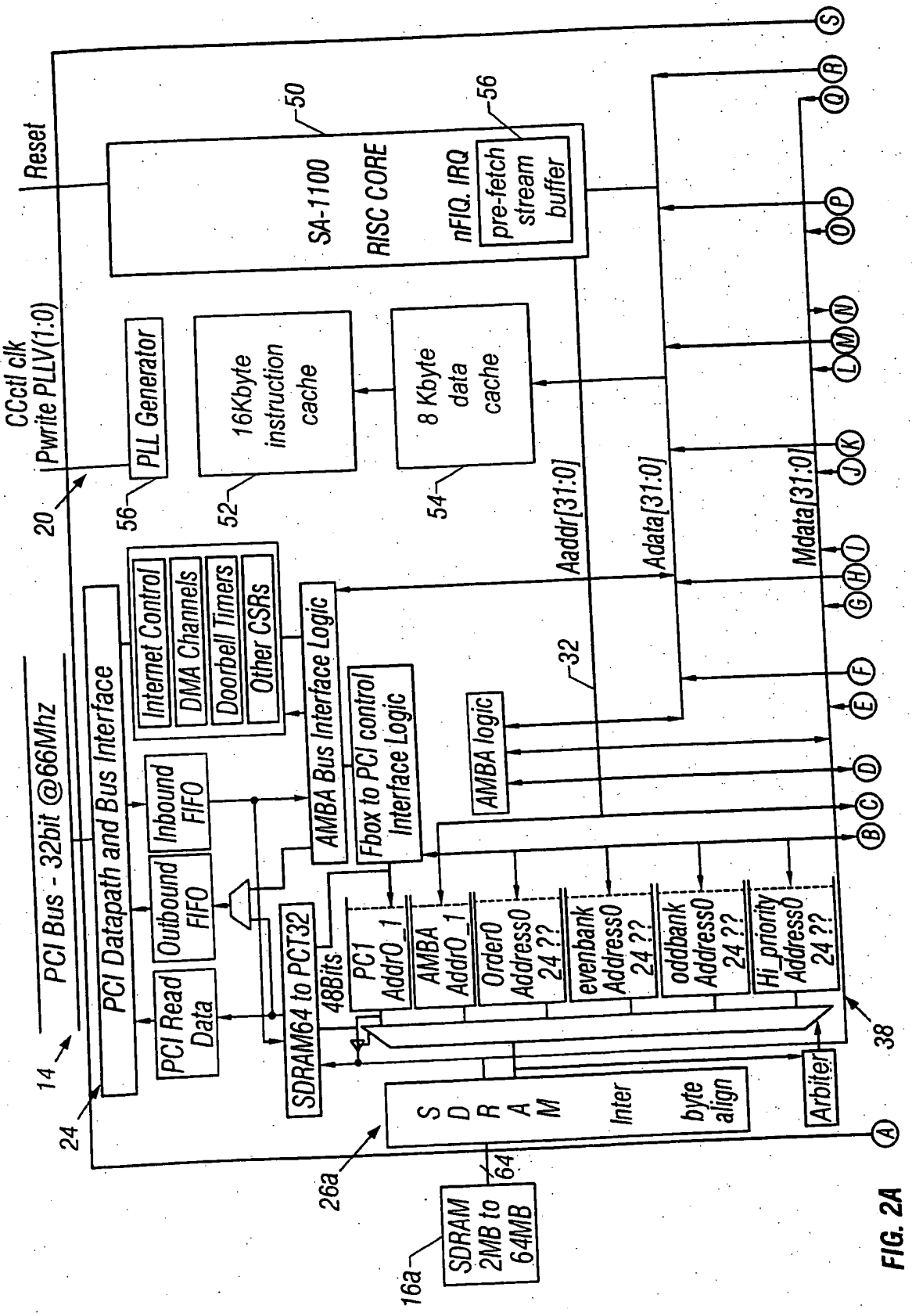

Reset
CCctl clk
Pwrite PLLV(1:0)
20
PLL Generator
56
50
SA-1100
RISC CORE
nFIQ. IRQ
56
pre-fetch stream buffer
16Kbyte instruction cache
52
8 Kbyte data cache
54
Aaddr[31:0]
Adata[31:0]
Mdata[31:0]
S
R
Q
P
N
M
L
K
J
I
H
G
F
E
D
C
B
A
32
Internet Control
DMA Channels
Doorbell Timers
Other CSRs
AMBA Bus Interface Logic
Fbox to PCI control Interface Logic
AMBA logic
PCI Bus - 32bit @66Mhz
PCI Datapath and Bus Interface
Inbound FIFO
Outbound FIFO
PCI Read Data
SDRAM64 to PCT32
48Bits
PCI Addr0_1
AMBA Addr0_1
Order0 Address0 24 ??
evenbank Address0 24 ??
oddbank Address0 24 ??
Hi_priority Address0 24 ??
14
24
38
SDRAM
Inter
byte align
Arbiter
26a
64
SDRAM 2MB to 64MB
16a

FIG. 2A

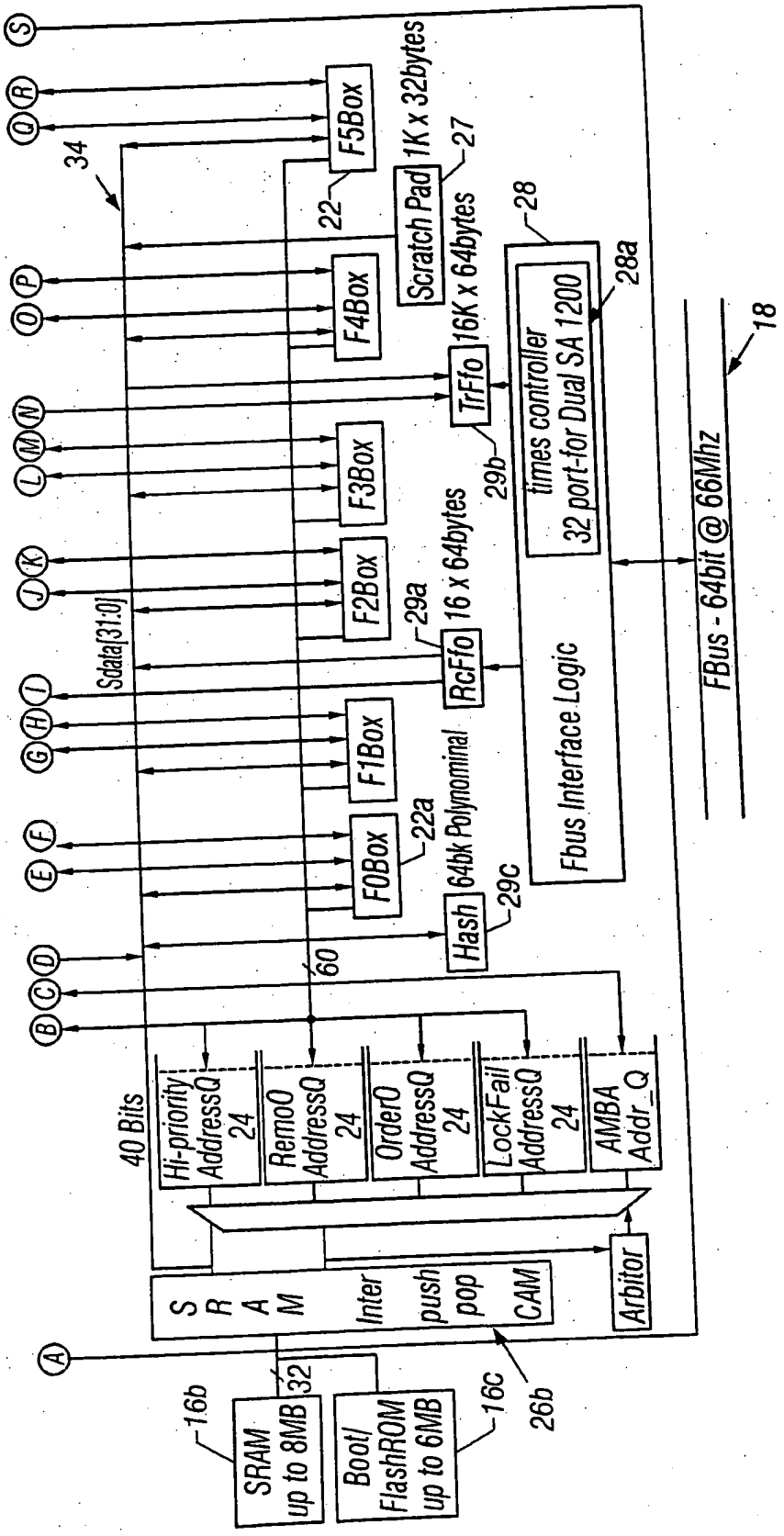

S
Q
R
P
O
N
M
L
K
J
I
H
G
F
E
D
C
B
A
34
F5Box
F4Box
F3Box
F2Box
F1Box
F0Box
22
22a
Scratch Pad 1K x 32bytes
27
TrFfo 16K x 64bytes
29b
28
28a
times controller
32 port-for Dual SA 1200
Fbus Interface Logic
RcFfo 16 x 64bytes
29a
Hash 64bk Polynominal
29c
Sdata[31:0]
60
40 Bits
Hi-priority AddressQ 24
RemoQ AddressQ 24
OrderQ AddressQ 24
LockFail AddressQ 24
AMBA Addr_Q
S R A M
Inter
push
pop
CAM
Arbitor
FBus - 64bit @ 66Mhz
18
16b
32
SRAM up to 8MB
Boot/ FlashROM up to 6MB
16c
26b

FIG. 2B

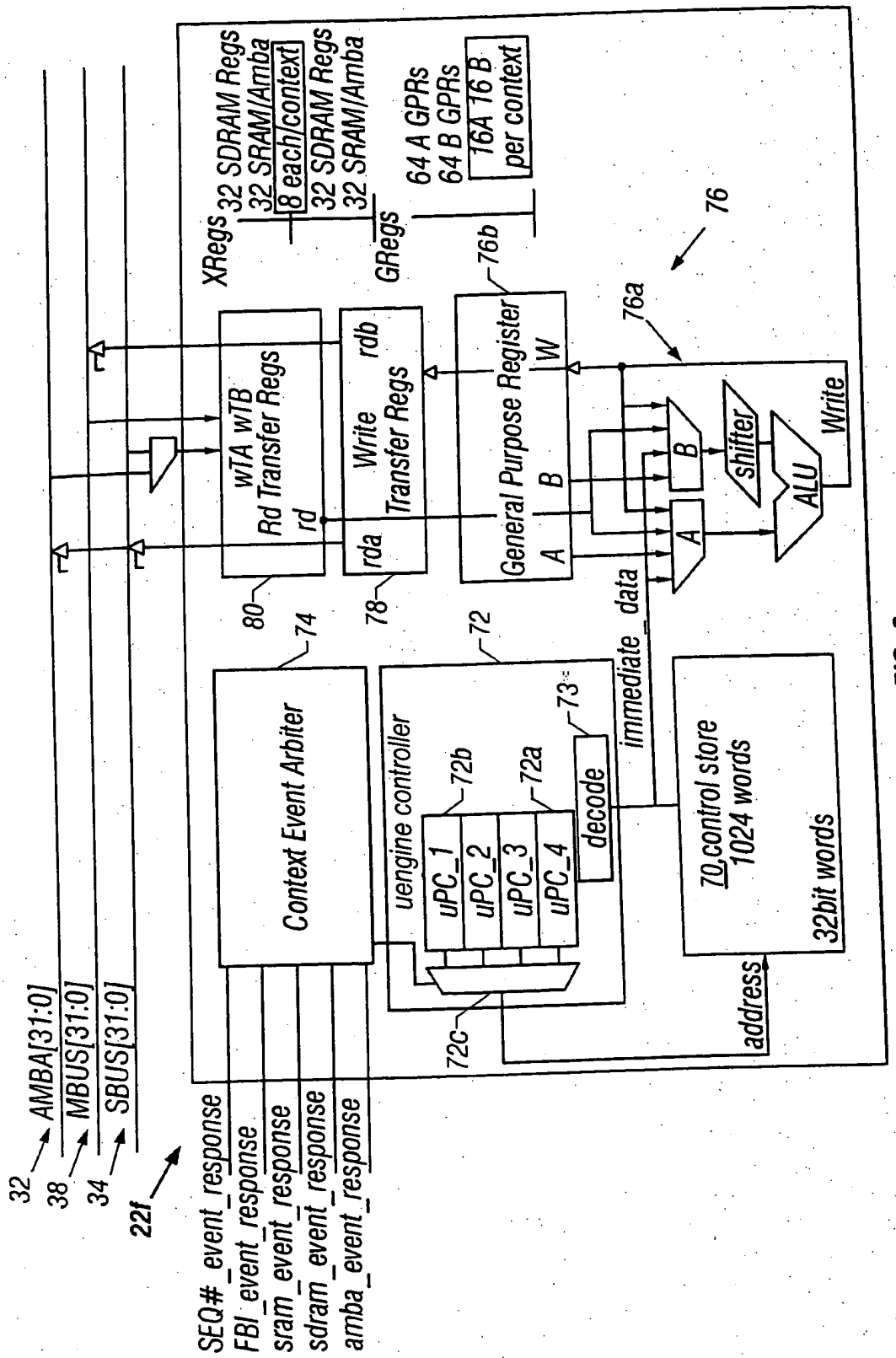

AMBA[31:0]
MBUS[31:0]
SBUS[31:0]
32
38
34
22f
SEQ#_event_response
FBI_event_response
sram_event_response
sdram_event_response
amba_event_response
Context Event Arbiter
74
uengine controller
72
72b
72a
73
uPC_1
uPC_2
uPC_3
uPC_4
decode
72c
immediate_data
70 control store
1024 words
32bit words
address
80
wTA wTB
Rd Transfer Regs
rd
78
rda
Write rdb
Transfer Regs
XRegs
32 SDRAM Regs
32 SRAM/Amba
8 each/context
GRegs
32 SDRAM Regs
32 SRAM/Amba
64 A GPRs
64 B GPRs
16A 16'B
per context
76b
General Purpose Register
W
A
B
76
76a
A
B
shifter
ALU
Write

FIG. 3

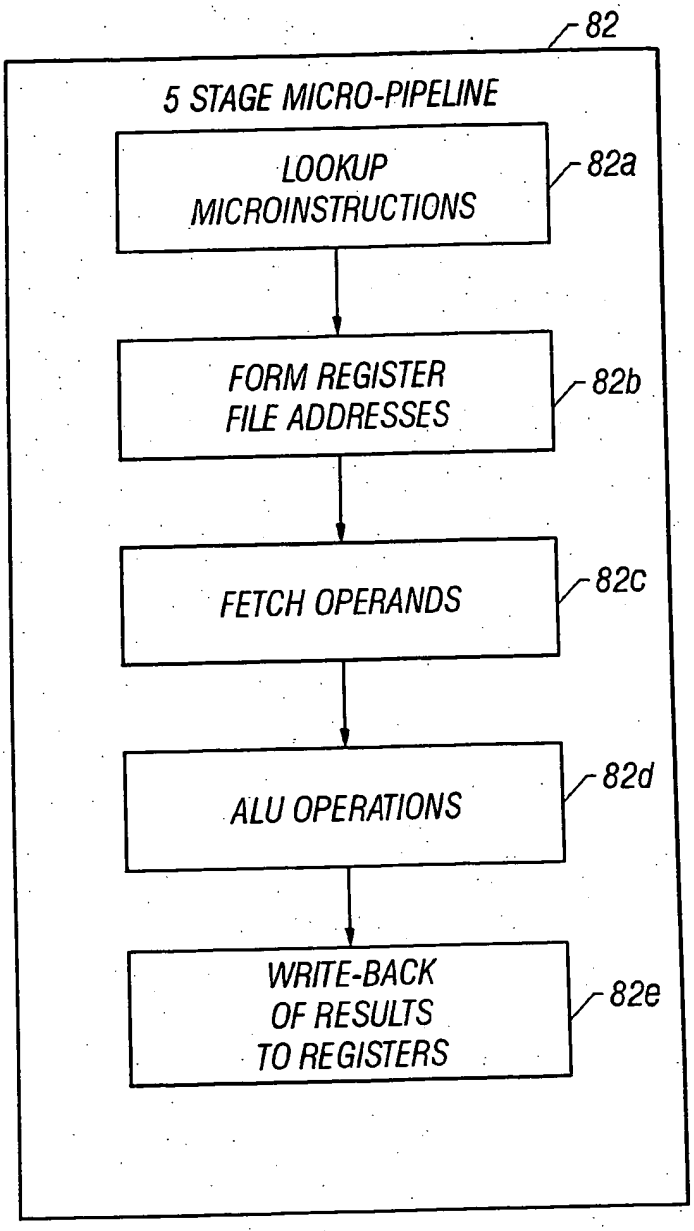
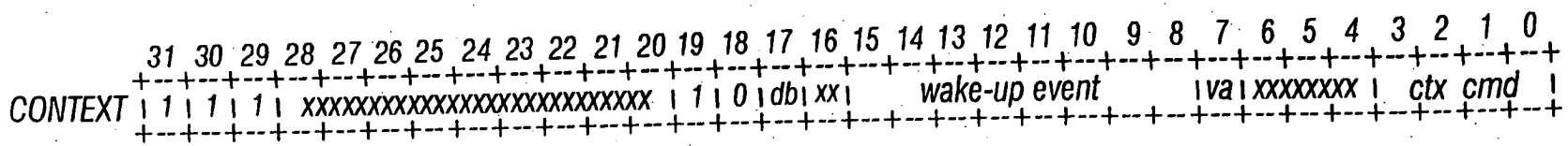

82
5 STAGE MICRO-PIPELINE
LOOKUP MICROINSTRUCTIONS
82a
FORM REGISTER FILE ADDRESSES
82b
FETCH OPERANDS
82c
ALU OPERATIONS
82d
WRITE-BACK OF RESULTS TO REGISTERS
82e

FIG. 3A



Context Descriptors:

1) Wake-up Events (Bits 8-15)

0 = kill
1 = voluntary
2 = SRAM
4 = SDRAM
8 = FBI
16 = INTER_THREAD
32 = PCI_DMA_1
64 = PCI_DMA_2
128 = SEQ_NUM_LSB

2) db -> branch defer amount (Bit 17)
3) va -> value of sequence number (Bit 7)
4) OPCODE Bits (29-31)
5) cxt_cmd

FIG. 3B

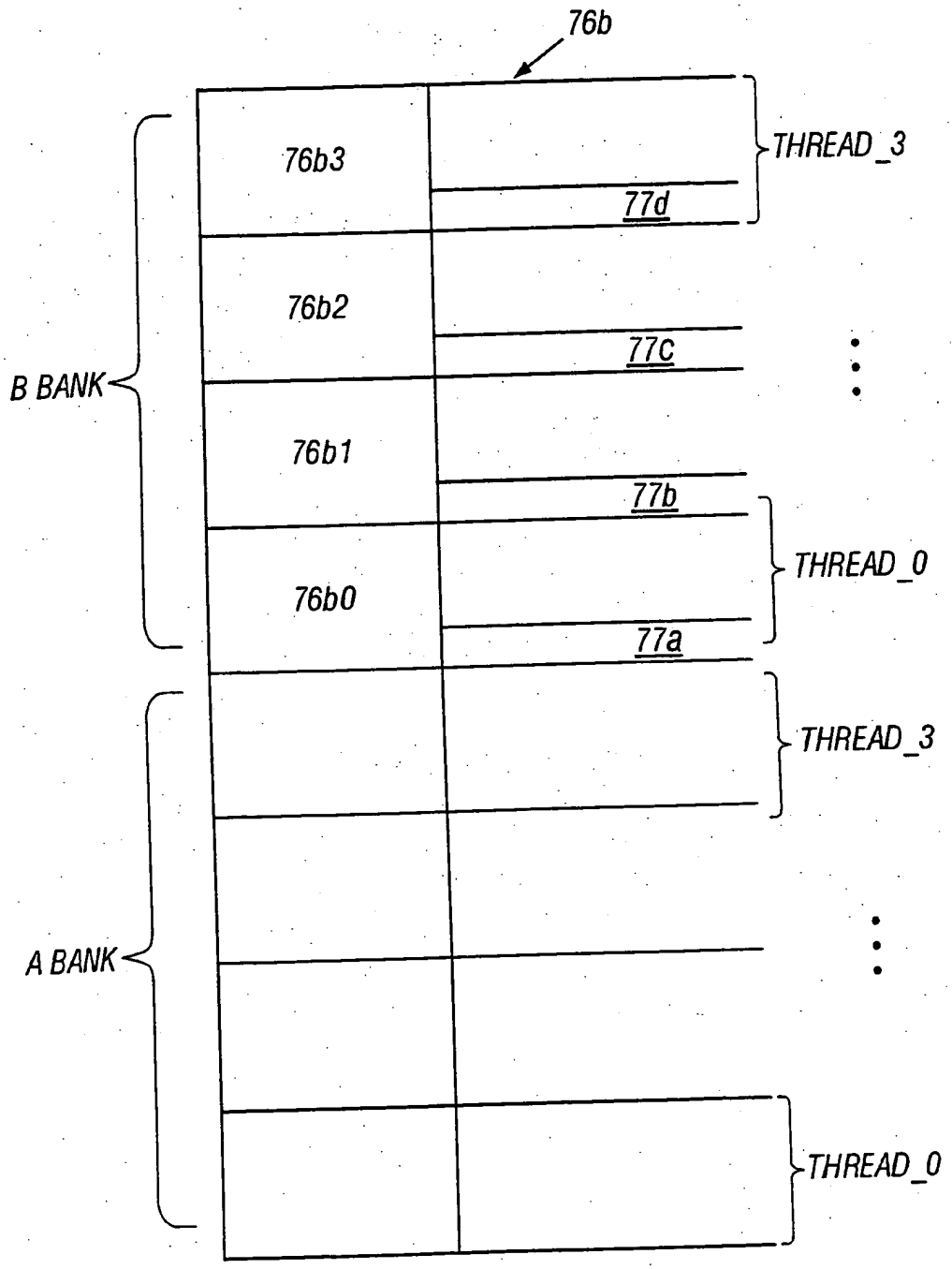

76b
76b3
77d
THREAD_3
76b2
77c
B BANK
76b1
77b
76b0
77a
THREAD_0
THREAD_3
A BANK
THREAD_0

FIG. 3C

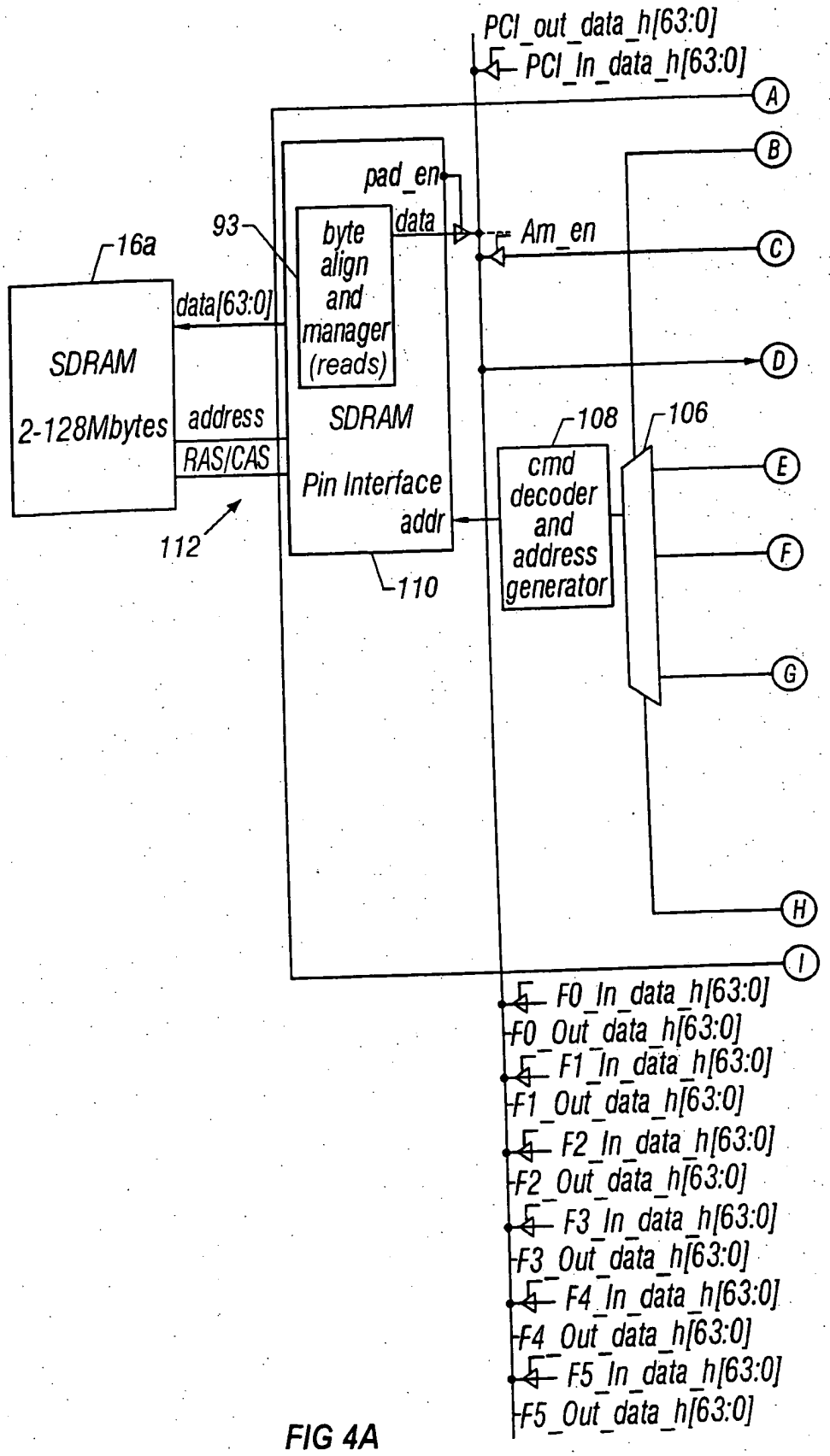

PCI_out_data_h[63:0]
PCI_In_data_h[63:0]
A
B
C
D
E
F
G
H
I
pad_en
data
Am_en
93
byte align and manager (reads)
16a
SDRAM
2-128Mbytes
data[63:0]
address
RAS/CAS
112
SDRAM
Pin Interface
addr
110
108
106
cmd decoder and address generator
F0_In_data_h[63:0]
F0_Out_data_h[63:0]
F1_In_data_h[63:0]
F1_Out_data_h[63:0]
F2_In_data_h[63:0]
F2_Out_data_h[63:0]
F3_In_data_h[63:0]
F3_Out_data_h[63:0]
F4_In_data_h[63:0]
F4_Out_data_h[63:0]
F5_In_data_h[63:0]
F5_Out_data_h[63:0]

FIG 4A

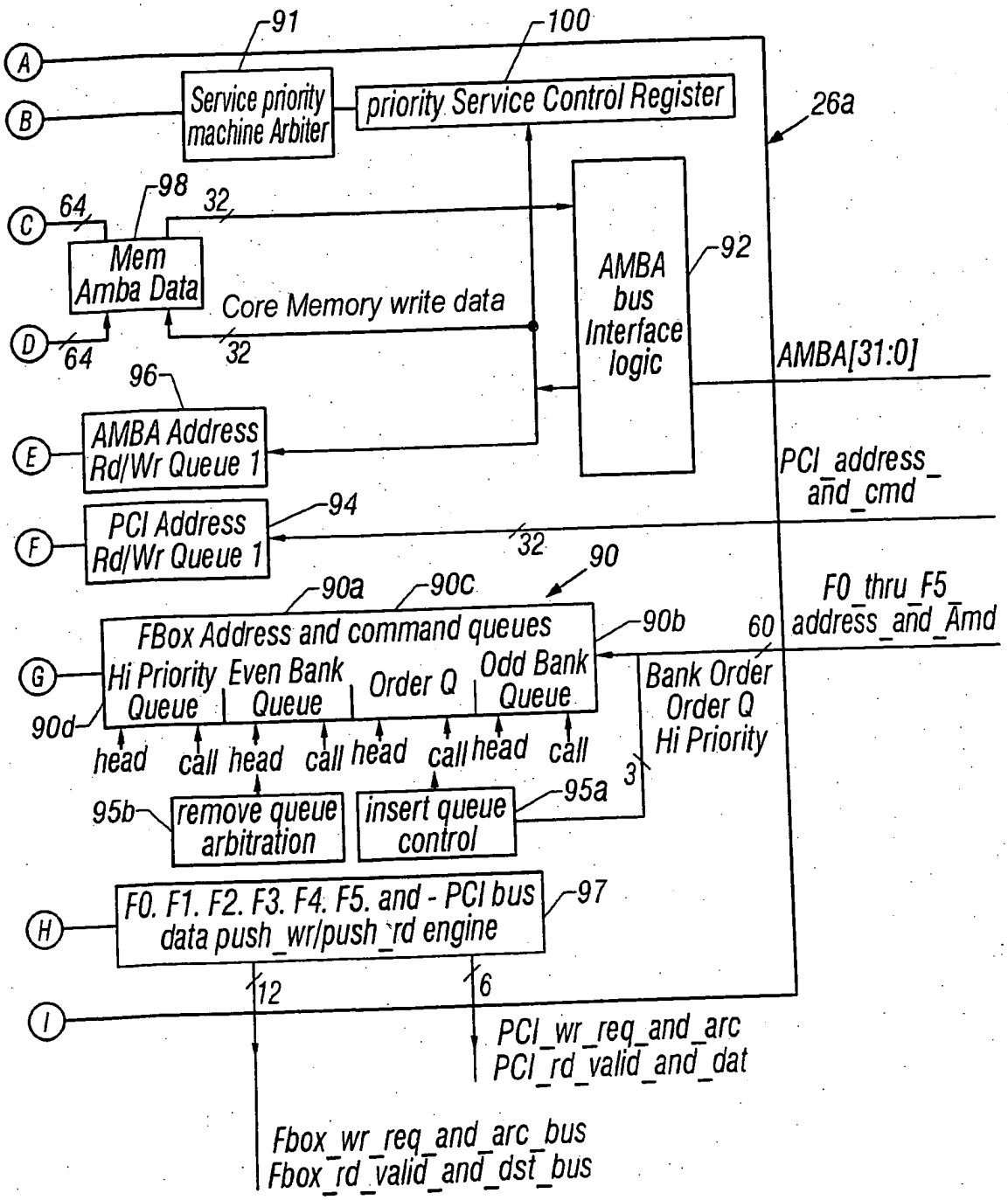

91
100
A
B
Service priority machine Arbiter
priority Service Control Register
26a
64
98
32
C
Mem Amba Data
Core Memory write data
AMBA bus Interface logic
92
D
64
32
AMBA[31:0]
96
E
AMBA Address Rd/Wr Queue 1
PCI_address_ and_cmd
F
PCI Address Rd/Wr Queue 1
94
32
90
90a
90c
F0_thru_F5 address_and_Amd
FBox Address and command queues
90b
60
G
Hi Priority Queue
Even Bank Queue
Order Q
Odd Bank Queue
Bank Order Order Q Hi Priority
90d
head
call
head
call
head
call
head
call
3
95b
remove queue arbitration
insert queue control
95a
H
F0. F1. F2. F3. F4. F5. and - PCI bus data push_wr/push_rd engine
97
12
6
I
PCI_wr_req_and_arc
PCI_rd_valid_and_dat
Fbox_wr_req_and_arc_bus
Fbox_rd_valid_and_dst_bus

FIG. 4B

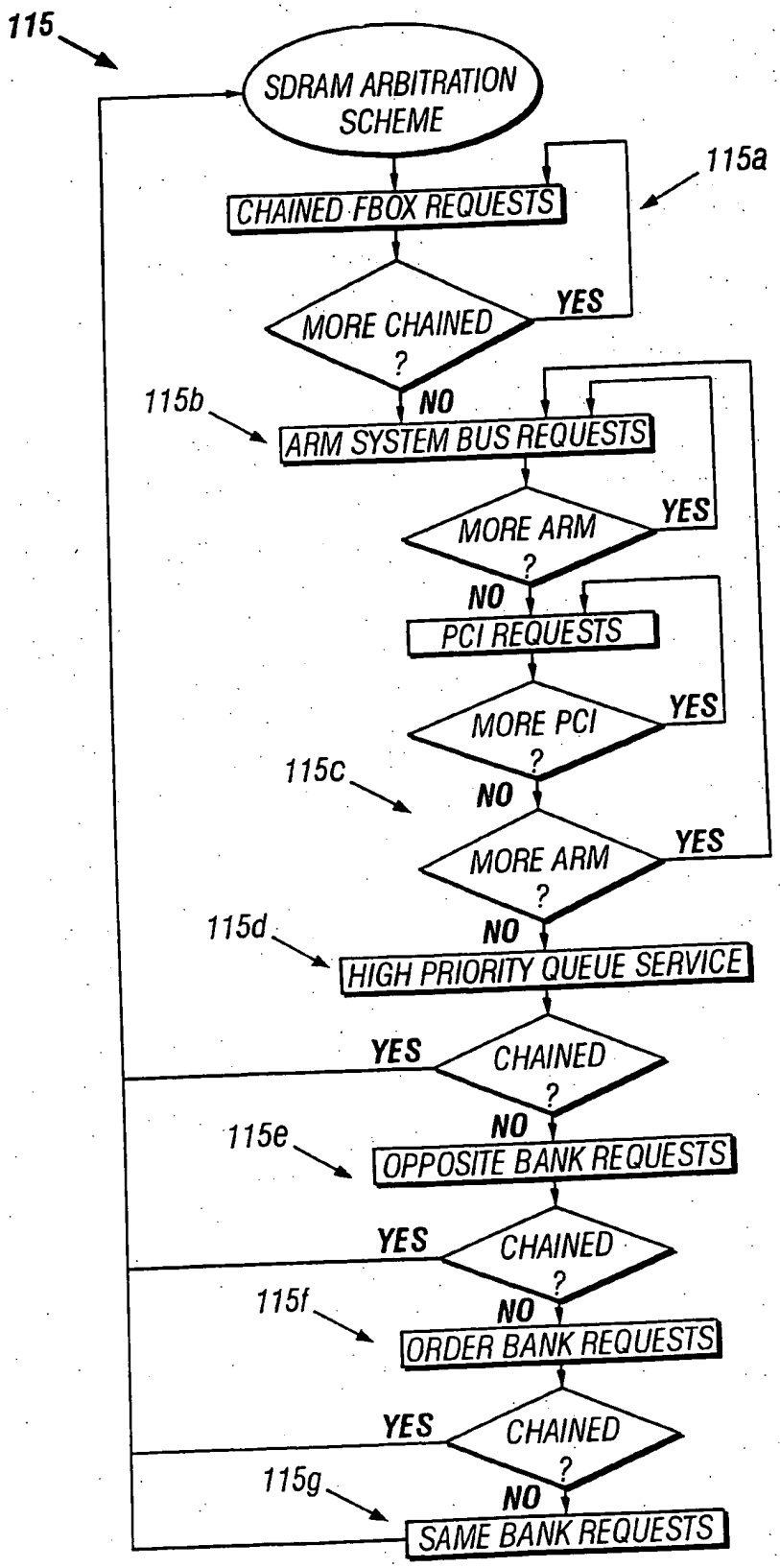

115
SDRAM ARBITRATION SCHEME
115a
CHAINED FBOX REQUESTS
MORE CHAINED ?
YES
NO
115b
ARM SYSTEM BUS REQUESTS
MORE ARM ?
YES
NO
PCI REQUESTS
MORE PCI ?
YES
115c
NO
MORE ARM ?
YES
NO
115d
HIGH PRIORITY QUEUE SERVICE
CHAINED ?
YES
NO
115e
OPPOSITE BANK REQUESTS
CHAINED ?
YES
NO
115f
ORDER BANK REQUESTS
CHAINED ?
YES
NO
115g
SAME BANK REQUESTS

FIG. 4C

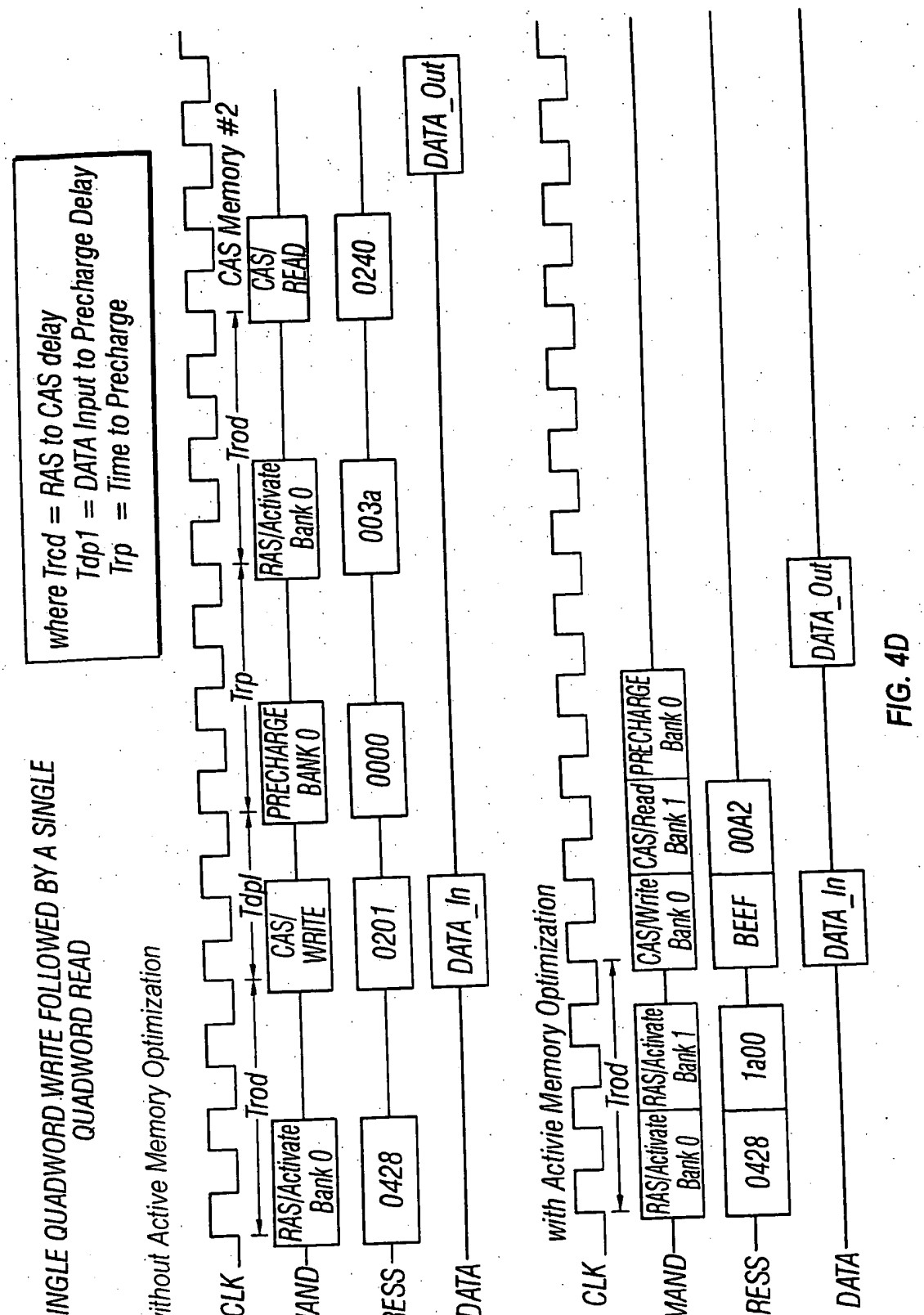

SINGLE QUADWORD WRITE FOLLOWED BY A SINGLE QUADWORD READ
without Active Memory Optimization
where Trcd = RAS to CAS delay
Tdp1 = DATA Input to Precharge Delay
Trp = Time to Precharge
CLK
COMMAND
ADDRESS
DATA
Trod
Tdp1
Trp
Trod
RAS/Activate Bank 0
CAS/ WRITE
PRECHARGE BANK 0
RAS/Activate Bank 0
CAS/ READ
CAS Memory #2
0428
0201
0000
003a
0240
DATA_In
DATA_Out
with Activie Memory Optimization
CLK
COMMAND
ADDRESS
DATA
Trod
RAS/Activate Bank 0
RAS/Activate Bank 1
CAS/Write Bank 0
CAS/Read Bank 1
PRECHARGE Bank 0
0428
1a00
BEEF
00A2
DATA_In
DATA_Out

FIG. 4D

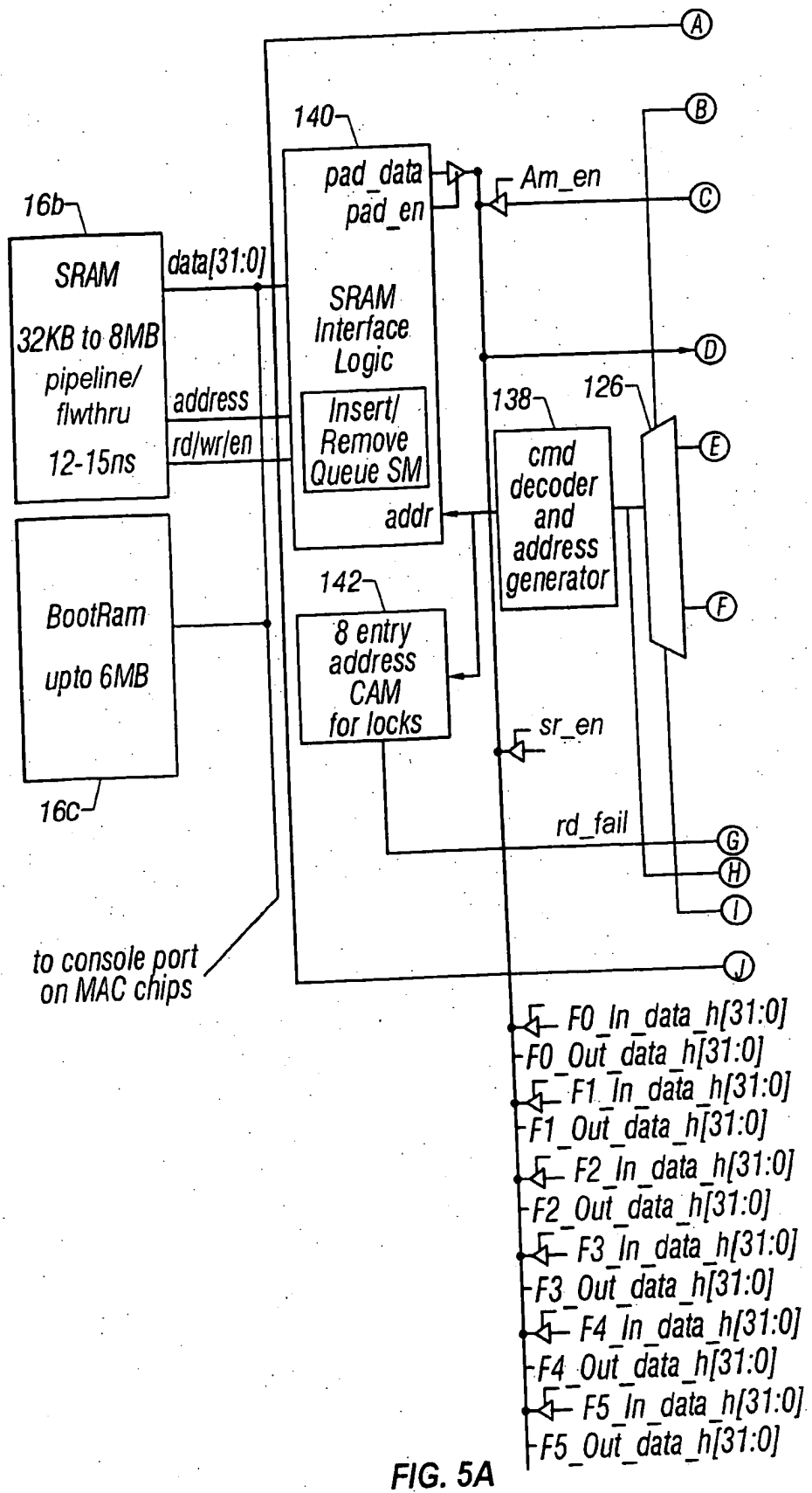

A
B
C
D
E
F
G
H
I
J
140
pad_data
pad_en
Am_en
16b
SRAM
32KB to 8MB
pipeline/
flwthru
12-15ns
data[31:0]
address
rd/wr/en
SRAM
Interface
Logic
Insert/
Remove
Queue SM
addr
138
126
cmd
decoder
and
address
generator
142
8 entry
address
CAM
for locks
BootRam
upto 6MB
16c
sr_en
rd_fail
to console port
on MAC chips
F0_In_data_h[31:0]
F0_Out_data_h[31:0]
F1_In_data_h[31:0]
F1_Out_data_h[31:0]
F2_In_data_h[31:0]
F2_Out_data_h[31:0]
F3_In_data_h[31:0]
F3_Out_data_h[31:0]
F4_In_data_h[31:0]
F4_Out_data_h[31:0]
F5_In_data_h[31:0]
F5_Out_data_h[31:0]

FIG. 5A

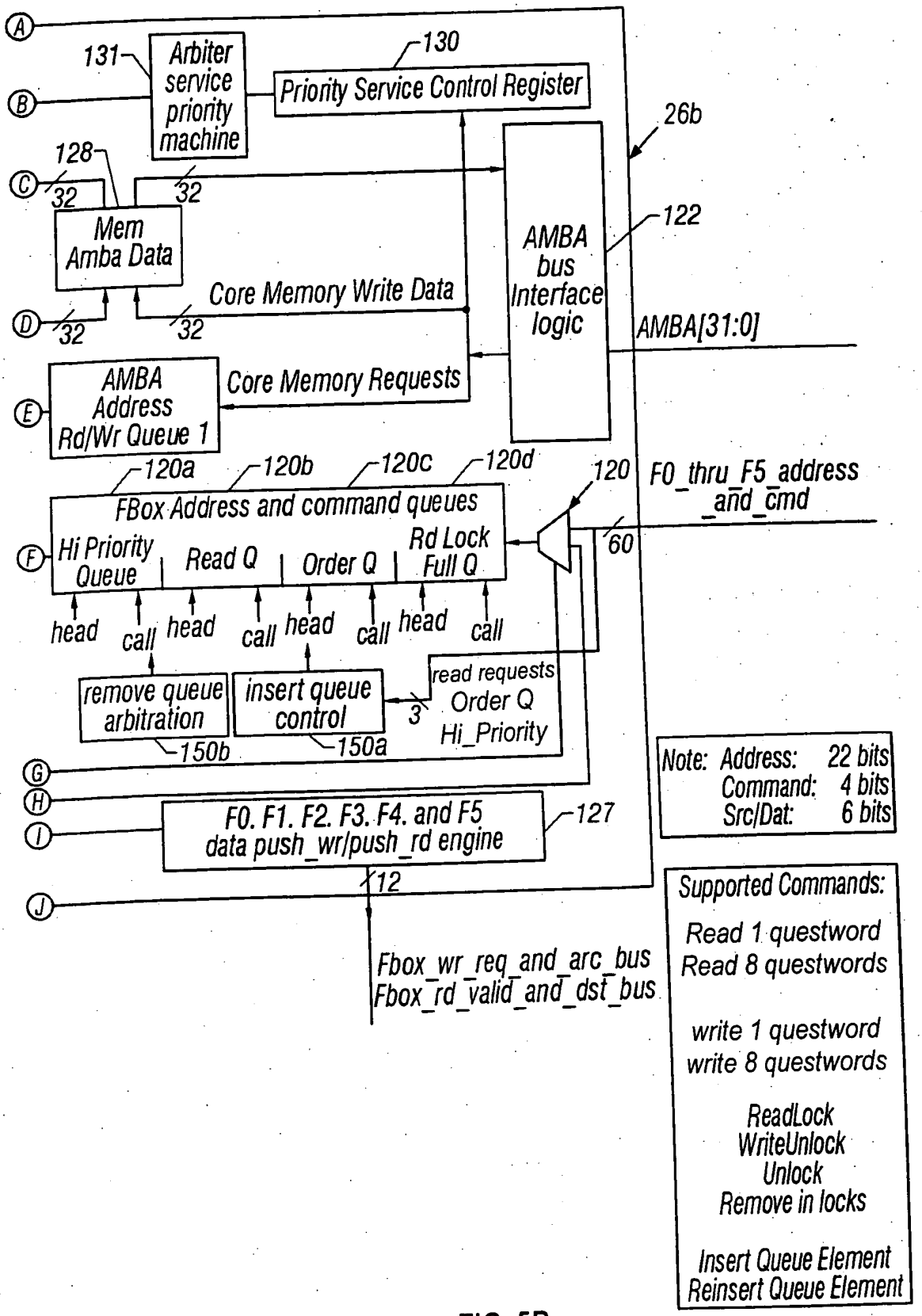

A
B
C
D
E
F
G
H
I
J
131
Arbiter service priority machine
130
Priority Service Control Register
26b
128
32
32
Mem Amba Data
122
AMBA bus Interface logic
Core Memory Write Data
32
32
AMBA[31:0]
AMBA Address Rd/Wr Queue 1
Core Memory Requests
120a
120b
120c
120d
120
F0_thru_F5_address _and_cmd
FBox Address and command queues
Hi Priority Queue
Read Q
Order Q
Rd Lock Full Q
60
head
call
head
call
head
call
head
call
remove queue arbitration
150b
insert queue control
150a
3
read requests Order Q Hi_Priority
F0. F1. F2. F3. F4. and F5 data push_wr/push_rd engine
127
12
Fbox_wr_req_and_arc_bus
Fbox_rd_valid_and_dst_bus
Note: Address: 22 bits
Command: 4 bits
Src/Dat: 6 bits
Supported Commands:
Read 1 questword
Read 8 questwords
write 1 questword
write 8 questwords
ReadLock
WriteUnlock
Unlock
Remove in locks
Insert Queue Element
Reinsert Queue Element

FIG. 5B

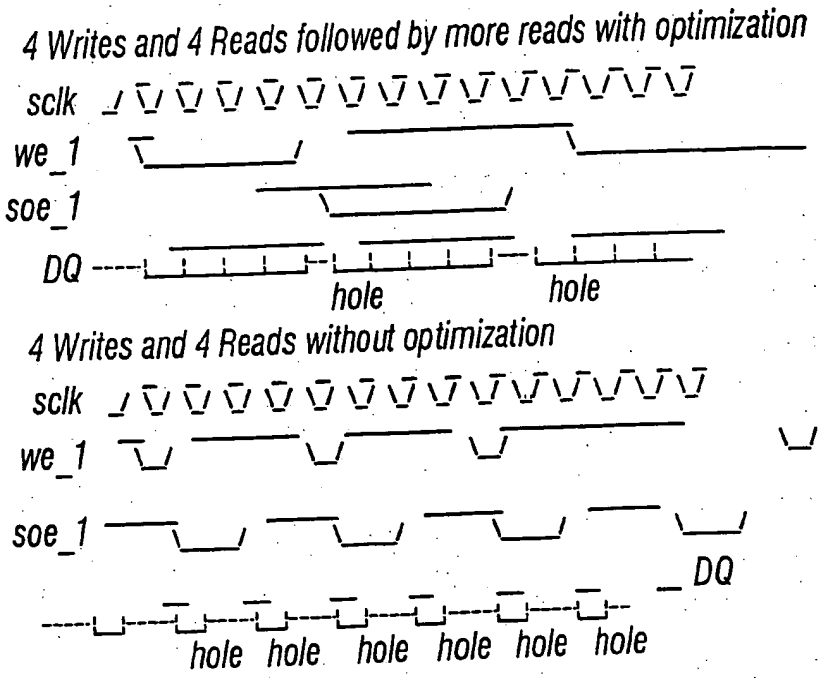

4 Writes and 4 Reads followed by more reads with optimization
sclk
we_1
soe_1
DQ
hole
hole
4 Writes and 4 Reads without optimization
sclk
we_1
soe_1
DQ
hole
hole
hole
hole
hole
hole

FIG. 5C

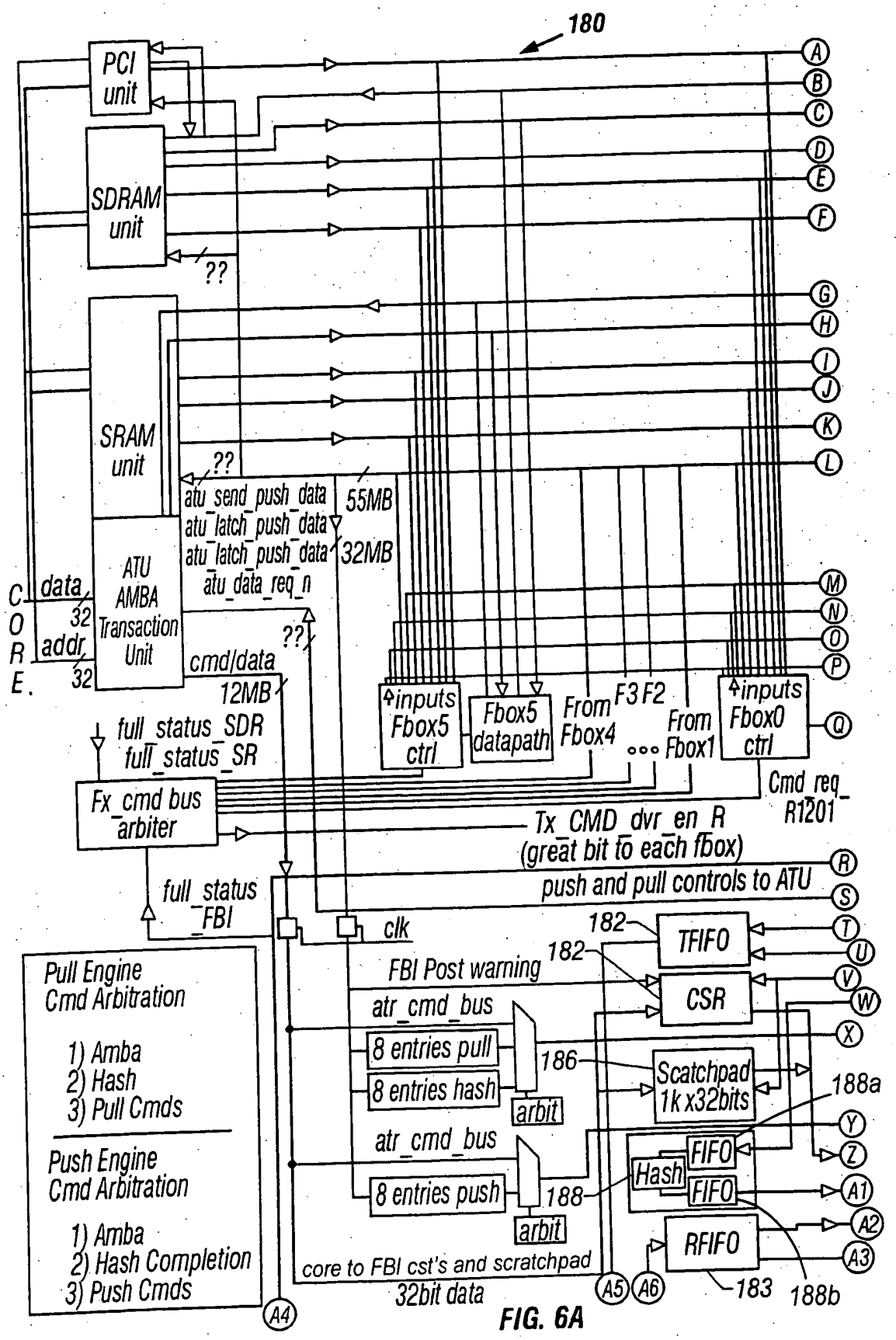

180
PCI unit
SDRAM unit
SRAM unit
ATU AMBA Transaction Unit
CORE
data 32
addr 32
??
atu_send_push_data
atu_latch_push_data
atu_latch_push_data
atu_data_req_n
55MB
32MB
cmd/data
12MB
full_status_SDR
full_status_SR
Fx_cmd bus arbiter
full_status FBI
clk
inputs Fbox5 ctrl
Fbox5 datapath
From Fbox4
F3 F2
From Fbox1
inputs Fbox0 ctrl
Cmd_req_R1201
Tx_CMD_dvr_en_R
(great bit to each fbox)
push and pull controls to ATU
Pull Engine Cmd Arbitration
1) Amba
2) Hash
3) Pull Cmds
Push Engine Cmd Arbitration
1) Amba
2) Hash Completion
3) Push Cmds
FBI Post warning
182
TFIFO
CSR
atr_cmd_bus
8 entries pull
8 entries hash
arbit
186
Scatchpad 1k x32bits
188a
atr_cmd_bus
8 entries push
arbit
188
Hash
FIFO
FIFO
RFIFO
183
188b
core to FBI cst's and scratchpad
32bit data
A1
A2
A3
A4
A5
A6
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

FIG. 6A

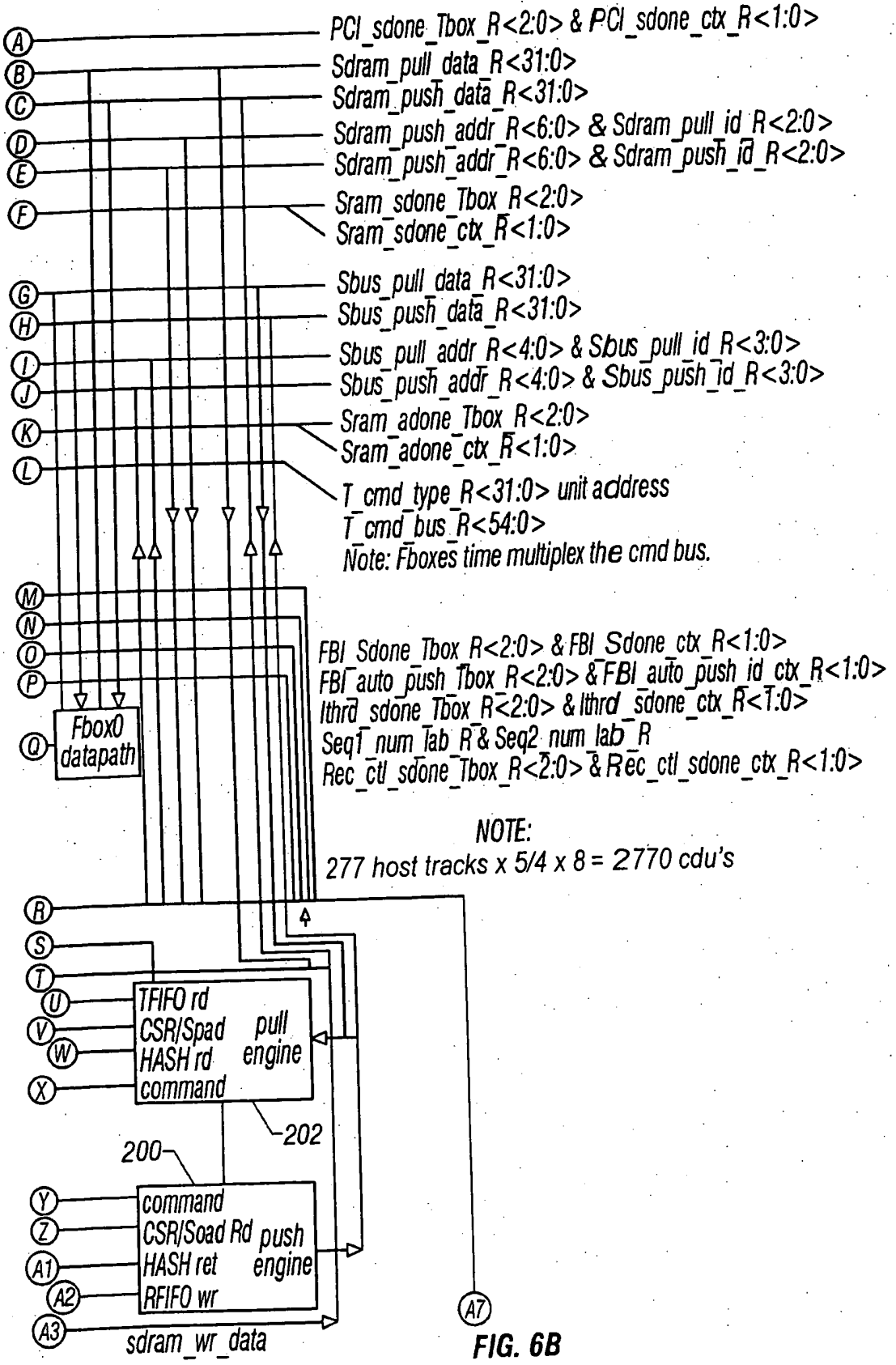

A
PCI_sdone_Tbox_R<2:0> & PCI_sdone_ctx_R<1:0>
B
Sdram_pull_data_R<31:0>
C
Sdram_push_data_R<31:0>
D
Sdram_push_addr_R<6:0> & Sdram_pull_id_R<2:0>
E
Sdram_push_addr_R<6:0> & Sdram_push_id_R<2:0>
F
Sram_sdone_Tbox_R<2:0>
Sram_sdone_ctx_R<1:0>
G
Sbus_pull_data_R<31:0>
H
Sbus_push_data_R<31:0>
I
Sbus_pull_addr_R<4:0> & Sbus_pull_id_R<3:0>
J
Sbus_push_addr_R<4:0> & Sbus_push_id_R<3:0>
K
Sram_adone_Tbox_R<2:0>
Sram_adone_ctx_R<1:0>
L
T_cmd_type_R<31:0> unit address
T_cmd_bus_R<54:0>
Note: Fboxes time multiplex the cmd bus.
M
N
O
P
FBI_Sdone_Tbox_R<2:0> & FBI_Sdone_ctx_R<1:0>
FBI_auto_push_Tbox_R<2:0> & FBI_auto_push_id_ctx_R<1:0>
Ithrd_sdone_Tbox_R<2:0> & Ithrd_sdone_ctx_R<1:0>
Seq1_num_lab_R & Seq2_num_lab_R
Rec_ctl_sdone_Tbox_R<2:0> & Rec_ctl_sdone_ctx_R<1:0>
Q
Fbox0 datapath
NOTE:
277 host tracks x 5/4 x 8 = 2770 cdu's
R
S
T
U
V
W
X
TFIFO rd
CSR/Spad
HASH rd
command
pull engine
202
200
Y
Z
A1
A2
A3
command
CSR/Soad Rd
HASH ret
RFIFO wr
push engine
sdram_wr_data
A7

FIG. 6B

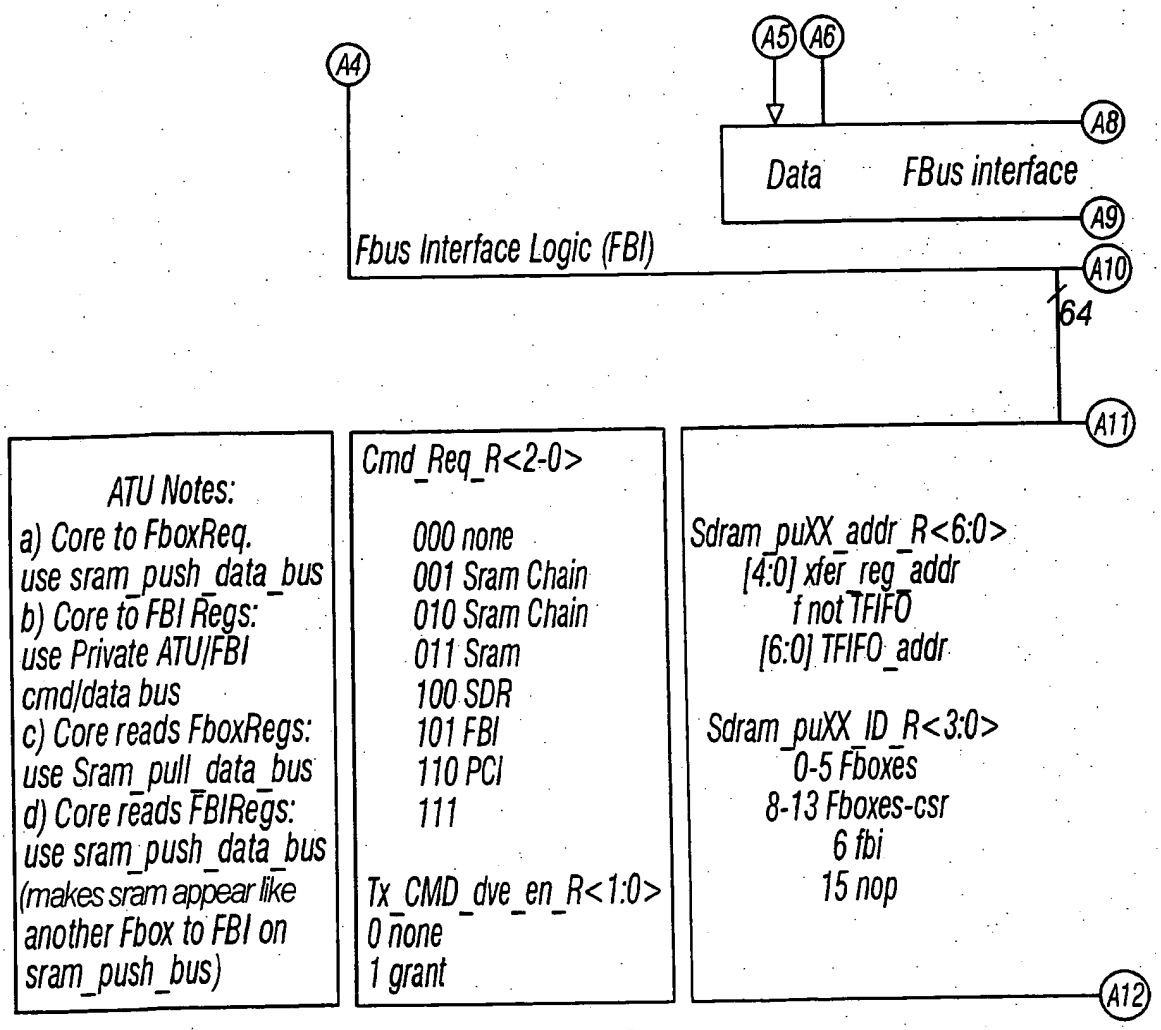

A4
A5
A6
A8
Data
FBus interface
A9
Fbus Interface Logic (FBI)
A10
64
A11
ATU Notes:
a) Core to FboxReq. use sram_push_data_bus
b) Core to FBI Regs: use Private ATU/FBI cmd/data bus
c) Core reads FboxRegs: use Sram_pull_data_bus
d) Core reads FBIRegs: use sram_push_data_bus
(makes sram appear like another Fbox to FBI on sram_push_bus)
Cmd_Req_R<2-0>
000 none
001 Sram Chain
010 Sram Chain
011 Sram
100 SDR
101 FBI
110 PCI
111
Tx_CMD_dve_en_R<1:0>
0 none
1 grant
Sdram_puXX_addr_R<6:0>
[4:0] xfer_reg_addr
f not TFIFO
[6:0] TFIFO_addr
Sdram_puXX_ID_R<3:0>
0-5 Fboxes
8-13 Fboxes-csr
6 fbi
15 nop
A12

FIG. 6C

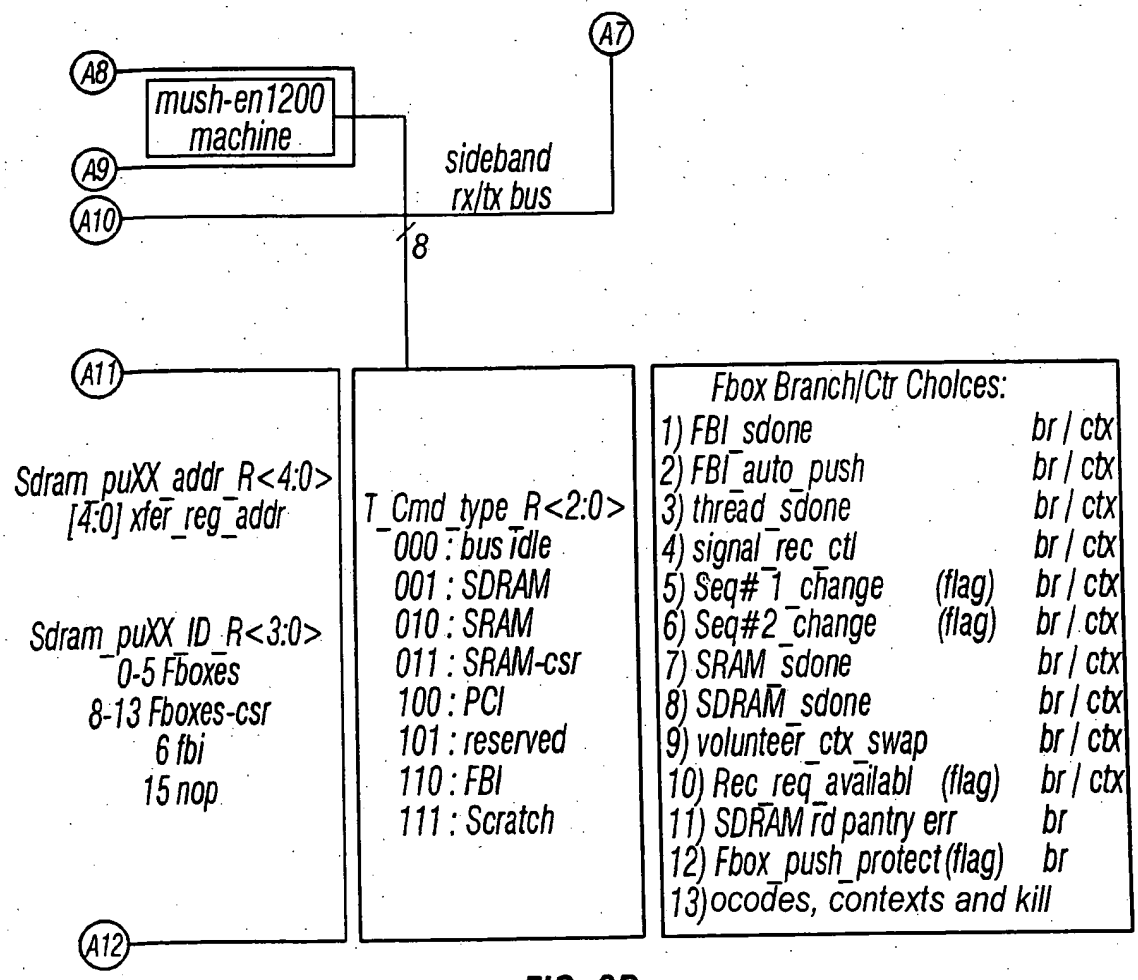

A7
A8
mush-en1200 machine
A9
A10
sideband rx/tx bus
8
A11
Sdram_puXX_addr_R<4:0>
[4:0] xfer_reg_addr
Sdram_puXX_ID_R<3:0>
0-5 Fboxes
8-13 Fboxes-csr
6 fbi
15 nop
A12
T_Cmd_type_R<2:0>
000 : bus idle
001 : SDRAM
010 : SRAM
011 : SRAM-csr
100 : PCI
101 : reserved
110 : FBI
111 : Scratch
Fbox Branch/Ctr Choices:
1) FBI_sdone br / ctx
2) FBI_auto_push br / ctx
3) thread_sdone br / ctx
4) signal_rec_ctl br / ctx
5) Seq#1_change (flag) br / ctx
6) Seq#2_change (flag) br / ctx
7) SRAM_sdone br / ctx
8) SDRAM_sdone br / ctx
9) volunteer_ctx_swap br / ctx
10) Rec_req_availabl (flag) br / ctx
11) SDRAM rd pantry err br
12) Fbox_push_protect (flag) br
13) ocodes, contexts and kill

FIG. 6D